Root Management

Delegated Votes

Root Leader

- Broadcast command to all replicas.
- Resolve conflicting delegations by comparing subquorum terms.
- If current vote count is a majority, begin epoch transition.

Root Delegates

- if epoch < current epoch: send no votes
- if vote undelegated: send self vote
- if candidate: send self vote
- if delegate: send all votes

Vote: (epoch **e**, quorum **q**, term **t**, votes **v**)

"Nuclear" Option

Delegations are only valid for the next epoch change. If enough delegates have failed that the epoch change cannot be made, a "nuclear" option resets delegates.

Triggered by a nuclear timeout $\gg$ root election timeout to ensure root leader is dead and delegates can't establish leader.

- Increment epoch beyond vote delegation limit, resetting all delegations.
- Conduct new root election/epoch change with all available replicas.
- Update health of all failed nodes and reconfigure epoch.

Epoch Decisions

Epoch Changes

Initiated by request, reconfiguration, localization, quiescence procedures.

Root Leader

- Monotonically increase epoch number, Define subquorums, initial leaders.
- Initiate delegated vote on epoch-change.
- On commit, begin fuzzy transition.

Subquorum Replicas

- Write tombstone into current log.
- Finalize commit for accesses prior to the tombstone record, forward new requests.
- On tombstone commit: truncate and archive log, join new subquorum configuration.

Fuzzy Transitions

Initiating: leader of subquorum in e-1
Remote: leader of subquorum in e

- Initiating sends last committed command for every object required by remote, Null for objects without accesses, and number of outstanding entries.
- Remote appends last entries and performs batch consensus to bring subquorum to the Same state.
- On remote commit, reports to root leader and begins accepting new accesses.

Note: background anti-entropy optimizes handoff process by reducing data volume.

Operations

Consensus and Accesses

Clients are forwarded to the subquorum leader with responsibility for requested object(s).

- **Read(o):** Leader responds with last committed entry; marks response if uncommitted entry for object exits. Adds read access to log but does not begin consensus (aggregates reads with writes).
- **Write(o):** Leader increments objects version number and creates a corresponding log entry. Sends consensus request and responds to client when the entry is committed.

Remote Accesses

In a multi-object transaction, remote accesses serialize inter-quorum access.

Initiating: append entries in log and send remote access request to remote leader.

Remote: create sub-epoch to demarcate remote access, add entry and respond to initiating replica when committed.

Initiating: on remote commit, create local sub-epoch, and commit entries appended to logs.